Improving Virtual Reality Streaming using HTTP/2

Stefano Petrangeli*
Filip De Turck
Ghent University - imec
name.surname@ugent.be

Viswanathan Swaminathan
Adobe Research
vishy@adobe.com

Mohammad Hosseini
University of Illinois at Urbana-Champaign
shossen2@illinois.edu

ABSTRACT

The demand for 360°Virtual Reality (VR) videos is expected to grow in the near future, thanks to the diffusion of VR headsets. VR Streaming is however challenged by the high bandwidth requirements of 360°videos. To save bandwidth, we spatially tile the video using the H.265 standard and stream only tiles in view at the highest quality. The video is also temporally segmented, so that each temporal segment is composed of several spatial tiles. In order to minimize quality transitions when the user moves, an algorithm is developed to predict where the user is likely going to watch in the near future. Consequently, predicted tiles are also streamed at the highest quality. Finally, the server push in HTTP/2 is used to deliver the tiled video. Only one request is sent from the client; all the tiles of a segment are automatically pushed from the server. This approach results in a better bandwidth utilization and video quality compared to traditional streaming over HTTP/1.1, where each tile has to be requested independently by the client. We showcase the benefits of our framework using a prototype developed on a Samsung Galaxy S7 and a Gear VR, which supports both tiled and non-tiled videos and streaming over HTTP/1.1 and HTTP/2. Under limited bandwidth conditions, we demonstrate how our framework can improve the quality watched by the user compared to a non-tiled solution where all of the video is streamed at the same quality. This result represents a major improvement for the efficient streaming of VR videos.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Human-centered computing** → **Virtual reality**; • **Networks** → *Network protocols*; *Public Internet*;

KEYWORDS

Virtual reality, HTTP Adaptive Streaming, HTTP/2, Server push, H.265

*This work was performed while the author was an intern at Adobe Research, San Jose (USA).

MMSys'17, Taipei, Taiwan
 978-1-4503-5002-0/17/06...$15.00
DOI: http://dx.doi.org/10.1145/3083187.3083224

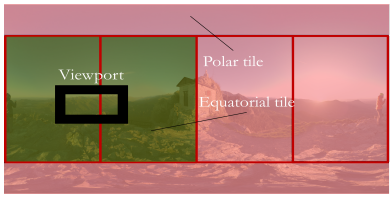

Figure 1: In tiled VR streaming, only tiles belonging to the viewport (in green) are streamed at the highest quality.

ACM Reference format:
Stefano Petrangeli, Viswanathan Swaminathan, Mohammad Hosseini, and Filip De Turck. 2017. Improving Virtual Reality Streaming using HTTP/2. In *Proceedings of MMSys'17, Taipei, Taiwan, June 20-23, 2017,* 4 pages.
DOI: http://dx.doi.org/10.1145/3083187.3083224

1 INTRODUCTION

Virtual Reality (VR) headsets are becoming extremely popular nowadays. Examples of VR headsets are the Facebook's Oculus Rift, Samsung's Gear VR, HTC's Vive and Google's Daydream, among others. Video streaming, which currently accounts for the largest portion of Internet traffic, will represent an important application for VR devices. Several forecasts report that VR videos are going to be worth more than 8 billion dollars by 2020[1]. Despite that, current VR streaming solutions are often affected by low video quality, due to the high bandwidth required to stream 360°VR videos.

Several approaches have been proposed to solve the aforementioned issue, with viewport-dependant solutions being good candidates. In viewport-dependent streaming, only the portion of the video watched through the viewport is streamed at the highest quality, while the rest of the video is streamed at a lower quality, to save bandwidth. In this scenario, multiple copies of the same VR video are created based on each possible position of the viewport, entailing high storage costs. As an example, several viewport-dependent streaming solutions require to store up to 30 different copies for each different video quality, for each 360°video [4]. To avoid additional storage costs while keeping the same benefits in terms of bandwidth reduction, spatial tiling can be used. In tiling, the VR video is divided into spatial regions, each encoded at multiple different quality levels. Besides being spatially tiled, the content is also temporally segmented. Each temporal segment is therefore composed of different video tiles. Only tiles belonging to the viewport are streamed at the highest quality (Figure 1). All the remaining tiles are streamed at a lower

[1] http://www.digitalstrategyconsulting.com/intelligence/2016/11/vr_video_to_surpass_video_gaming_by_2019_ovum.php

quality to save bandwidth. Unfortunately, tiling the video causes a significant increase in the number of requests when HTTP Adaptive Streaming (HAS), the de-facto standard for video streaming over the Internet, is used over HTTP/1.1. Each tile has to be retrieved independently and sequentially to create a complete temporal segment, meaning that several RTTs are lost to download a single segment.

We solved the aforementioned problems by developing a new framework for the efficient streaming of VR videos. First, the H.265 standard is used to encode the video, as it natively supports tiling and allows to decode the tiled video using a single decoder. This aspect is particularly important for resource-constrained devices, as mobile devices. Second, we use the server push feature of the HTTP/2 Standard to deliver the video [12]. In server push, a single request is sent from the client to request the tiles a segment is composed of. Based on this request, the server can push all the tiles back to back, thus eliminating the request overhead due to tiling. HTTP/2 shares the same methods, status codes and semantics with HTTP/1.1, entailing complete backward compatibility. Server push, and more generally HTTP/2, is also completely cache- and CDN-friendly. Third, a client-based viewport prediction algorithm is used to minimize the quality transitions when the user moves. By using prediction, we can foresee where a user is going to watch in the near future, and download in advance the right portion of the video at the highest quality. A complete proof-of-concept has been developed to showcase the benefits brought by the proposed framework. The demo has been implemented on a Samsung Galaxy S7 and a Gear VR. It allows to play both tiled and non-tiled videos, and to implement different viewport prediction algorithms. Also, the prototype supports adaptive streaming over HTTP/1.1, HTTP/2 and HTTP/2 server push, making it an ideal tool to experiment with different VR streaming solutions.

The remainder of this paper is structured as follows. Section 2 presents the related work on 360°VR video streaming and HTTP/2-based adaptive streaming. Section 3 describes in detail the proposed framework from an architectural point of view, while Section 4 reports the proof-of-concept setup. Section 5 concludes the paper.

2 RELATED WORK

2.1 360°Video Streaming

D'Acunto et al. propose a tiled solution to optimize the delivery of zoomable videos, which are affected by the same bandwidth problem as VR videos [3]. Tiling has also been used to deliver panoramic videos, often combined with viewport prediction to improve user experience during viewpoint changes [5, 8]. In these works, the video is tiled using H.264, which does not natively support tiling. This aspect complicates the synchronization of the tiles, as each tile has to be decoded independently. Le Feuvre et al. use H.265 to spatially divide the 360°video and gradually degrade the quality outside the viewport [7]. A VR framework has been proposed where only the portion of the 360°video watched by the user is actually transmitted, to save bandwidth [11]. The developed viewport prediction algorithm should therefore be extremely precise in order to avoid stalling when the viewport changes. While all of the above research is successful in addressing some of the problems affecting VR streaming, there is no single solution to address all problems. Our work is an attempt to provide a comprehensive solution for VR streaming. By using H.265 to tile the video, we avoid the synchronization issue at client side introduced by tiling. The video is transported using HTTP/2, which eliminates the significant increase of requests due to tiling. Finally, viewport prediction can reduce the quality degradation introduced by assigning lower qualities for tiles outside the viewport, by anticipating user's movements.

Budagavi et al. optimize the encoding process to reduce the bit-rate of VR videos, by gradually smoothing the quality of the bottom and top part of an equirectangular projection [1]. Prior work has proposed a new tiling structure for 360°video, potentially saving up to 30% of the bandwidth compared to a non-tiled video [6]. Our approach can be considered complimentary to these works. We mostly focus on the delivery of the video, rather that its encoding and preparation, using the server push feature of HTTP/2 and viewport prediction. A new prototype has been developed to showcase the benefits of the proposed approach.

2.2 HTTP/2-Based Adaptive Streaming

Wei et al. are the first to investigate how server push can improve the delivery of HAS streams [12]. They decrease the camera-to-display delay, by reducing the segment duration and pushing k segments after a single HTTP GET request is issued by the client. The k-push has also been extended to optimize the battery lifetime on mobile devices [13]. Cherif et al. use server push and WebSocket to reduce the startup delay in a DASH streaming session [2]. In this work, we exploit the server push functionality to reduce the network overhead introduced by spatially tiling the video. Instead of pushing the segments one after other, we use the k-push mechanism to push the tiles composing a single video segment back to back from the server to the client.

3 VR STREAMING FRAMEWORK

Our VR streaming framework has three components, which we briefly describe in this section. First, the H.265 standard is used to tile the video. Second, a client-based algorithm decides the quality of each tile, by taking into account both the current and the predicted viewport. Third, the server push feature of HTTP/2 is used to deliver the video.

H.265 natively supports tiling and is therefore the best tool to prepare content for VR streaming [9]. Particularly, tiles are independent objects that can be requested independently by the client, even at different video qualities. At client-side, a single H.265 decoder is able to decode all the different tiles and provide a seamless playout. This aspect also avoids any synchronization issue between the different tiles, as a single decoder is involved in the decoding process.

While H.265 is used to encode the content, the quality of the different tiles is fully determined by the client. This decision is based both on the available network bandwidth and the position of the current and predicted viewport. In

this work, we consider as viewport the region with center the fixation point and 60-degrees in radius, known as the mid-peripheral region of the human eye. The quality of the tiles falling inside the current and predicted viewport should always be maximized to provide the best user experience. The remaining tiles can be streamed at a lower quality to save bandwidth. Estimating the future user viewport is fundamental in order to minimize quality transitions when the user moves. In this work, we use a speed-based approach, where the position of the future viewport is computed based on the position and speed of the current fixation point. We first obtain the position p at instant k of the current fixation point on the underlying 2D projection of the VR video, for example, latitude and longitude for an equirectangular projection. The future fixation point, which defines the future viewport, is computed as $p(k+\Delta) = p(k)+\Delta \times \widehat{p(k)}$, where Δ is the viewport prediction horizon and $\widehat{p(k)}$ is the fixation point speed. Once viewport tiles have been identified (i.e., tiles belonging to the current and future viewport), the actual quality is selected based on the available bandwidth. Assuming B is the available bandwidth and n_T is the total number of tiles, we first assign the lowest bit-rate to all the tiles, to guarantee that the whole video is streamed to the user, and update the bandwidth budget $B_{bdg} = B - n_T \times b_{low}$. Next, we assign the highest possible bit-rate b_{vw} to the n_{vw} viewport tiles, such that $b_{vw} \leq B_{bdg}/n_{vw}$. We then update the bandwidth budget (given by $B_{bdg} - b_{vw} \times n_{vw}$) and repeat the allocation for those tiles immediately outside the viewport. The same process is then repeated for the remaining tiles till we run out of bandwidth. This way, we mitigate the edge effect between tiles at different qualities by gradually reducing the quality as we move out of the viewport.

Once the tiles quality is decided, the client issues an HTTP GET request to the server. In classical HTTP/1.1, each tile should be requested independently and sequentially. This behavior entails that n_T RTTs are lost to retrieve a single temporal segment, which would lower the achieved throughput in mobile, high RTT networks. In our framework instead, a single GET request is issued by the client, which specifies the quality of each tile. Using the server push feature of HTTP/2, the server is able to push automatically and sequentially all the video tiles to the client. Particularly, we use the k-push approach proposed by Wei et al. [12], with k set to n_T. This approach eliminates the request overhead due to tiling and results in a better bandwidth utilization, even in high RTT networks. It is worth stressing that the k-push mechanism does not require any client status to be kept on the server. A push directive, as standardized by part 6 of the MPEG-DASH standard [10], embedded in the client request specifies the tiles qualities. We extended the push directive in a compatible way to allow the client specifying the tiles qualities. By parsing this directive, the server understands which tiles to push. HTTP/2 server push is by design cache compatible and several CDNs are starting to deploy it[2].

[2]https://blogs.akamai.com/2016/04/are-you-ready-for-http2-server-push.html

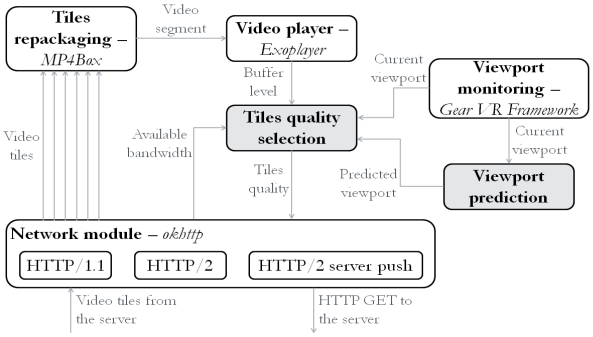

Figure 2: Illustrative diagram of the developed prototype. Gray boxes indicate algorithmic components. In italics, the names of the used libraries.

4 PROOF-OF-CONCEPT SETUP

The proposed framework has been implemented as a prototype on a Samsung Galaxy S7 and a Gear VR. A high-level description of the prototype is given in Figure 2. The Gear VR Framework[3] allows to develop VR applications on Android devices and provides general VR functionalities. The framework is mainly used in the *Viewport monitoring* module, as it allows to capture where the user is watching and to enable viewport-awareness. The *Tiles quality selection* module selects the quality of the tiles and takes as input: (i) the buffer level, (ii) the available bandwidth, (iii) the current viewport and (iv) the predicted viewport, computed by the *Viewport prediction* module. The tiles quality is then communicated to the *Network module*, implemented using the okhttp[4] library, which takes care of the actual streaming of the video segments. Both regular HTTP/1.1 and HTTP/2 protocols are supported. We also extended the okhttp library to support the server push functionality of HTTP/2. Once the tiles are downloaded from the server, the *Tiles repackaging* module, realized using the MP4Box[5] library, pre-process them before they are actually played by the *Video player*. This step is necessary because the ExoPlayer[6], which implements the video playout, is not able to directly play tiled videos. Particularly, tiles are concatenated into a single mp4 file using the *cat* command, and the raw HEVC stream is extracted using the *raw* command. Future versions of the ExoPlayer would allow to eliminate this step. Despite this process, the latency added to the system is less than 100 ms.

The VR video streamed by the client is the *Alba 360° Timelapse*, which was re-encoded into three quality levels corresponding to QP values equal to 30, 25 and 20 and bit-rates 1.6 Mbps, 3.2 Mbps and 7.1 Mbps. The segment duration is equal to 2 seconds. The encoding has been carried out using the HM encoder[7] (version 16.4), the reference software for H.265. HM supports motion-constrained encoding and allows to decode the tiled video with a single decoder at client-side.

[3]https://resources.samsungdevelopers.com/Gear_VR/020_GearVR_Framework_Project
[4]http://square.github.io/okhttp/
[5]https://gpac.wp.mines-telecom.fr/mp4box/
[6]https://developer.android.com/guide/topics/media/exoplayer.html
[7]https://hevc.hhi.fraunhofer.de/

Figure 3: Evolution of the viewport quality during viewport changes. Full black frames contain high quality regions, dashed frames low quality regions.

The video is divided into six tiles: two *polar* tiles and *four* equatorial tiles [6], as in Figure 1. This tiling structure is not natively supported by H.265, which only allows to tile the video into a regular grid, where all rows have the same amount of columns and vice-versa. Therefore, we concatenate all the tiles belonging to the polar region together, so that they can be requested as a single polar tile by the client. It is worth stressing that our prototype is independent of the actual tiling structure of the video, so that any H.265-compliant tiling structure can be used instead of that adopted in this paper. The HTTP server, where the VR video is hosted, is implemented via a Jetty server, which was extended to implement the HTTP/2 k-push functionality [13]. In our prototype, k corresponds to the number of tiles.

A 5Ghz ad-hoc wireless network connects the Samsung S7 to a MacBook Pro Retina, where the Jetty server is located. The network bandwidth and RTT are fixed to 5 Mbps and 100 ms, respectively. This configuration allows to clearly highlight the benefits of the proposed framework, which will be showcased using the developed prototype. The users will be able to test the prototype in different VR streaming configurations. First, the user will experience the quality that can be obtained using a non-tiled approach where the whole video is streamed at the same quality. Given the bandwidth limitation, only the second quality can be streamed in this case. Second, a tiled approach over HTTP/1.1 will be shown. Despite tiling, also in this case the highest quality cannot be reached due to the high RTT. Finally, the proposed tiled approach over HTTP/2 server push will be showcased, with and without viewport prediction. Figure 3 provides a snapshot of the VR video, recorded using the developed prototype, representing how the viewport quality evolves while the user explores the VR video. The importance of viewport prediction on the viewed quality will also be evident as the user moves around the video.

5 CONCLUSIONS

In this paper, we presented a proof-of-concept for the efficient streaming of VR videos, developed on a Samsung Galaxy S7 and a Gear VR. The developed prototype stream VR videos tiled using the H.265 standard, where only tiles belonging to the viewport are at the highest quality, to save bandwidth. Moreover, to provide a graceful transition during viewport changes, a prediction algorithm can foresee which tiles are going to be watched in the near future, so that they can be requested in advance at the highest quality as well. The HTTP/2 server push is used to efficiently deliver the video over the best-effort Internet. The client only needs to send a single HTTP GET request specifying the quality of the tiles, which are then pushed by the server. The gains showcased by the presented prototype represent an important step toward the efficient streaming of VR videos with consistent quality.

REFERENCES

[1] M. Budagavi et al. 360 degrees video coding using region adaptive smoothing. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 750–754, 2015.

[2] W. Cherif et al. Dash fast start using http/2. In *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 25–30, 2015.

[3] L. D'Acunto et al. Using mpeg dash srd for zoomable and navigable video. In *Proceedings of the 7th International Conference on Multimedia Systems*, pages 34:1–34:4, 2016.

[4] Facebook. Next-generation video encoding techniques for 360 video and vr. https://code.facebook.com/posts/1126354007399553/next-generation-video-encoding-techniques-for-360-video-and-vr/.

[5] V. R. Gaddam et al. Tiling in interactive panoramic video: Approaches and evaluation. *IEEE Transactions on Multimedia*, 18(9):1819–1831, 2016.

[6] M. Hosseini et al. Adaptive 360 vr video streaming: Divide and conquer! In *Proceedings of the IEEE International Symposium on Multimedia*, 2016.

[7] J. Le Feuvre et al. Tiled-based adaptive streaming using mpeg-dash. In *Proceedings of the 7th International Conference on Multimedia Systems*, pages 41:1–41:3, 2016.

[8] S. Y. Lim et al. Tiled panoramic video transmission system based on mpeg-dash. In *2015 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 719–721, 2015.

[9] K. Misra et al. An overview of tiles in hevc. *IEEE Journal of Selected Topics in Signal Processing*, 7(6):969–977, 2013.

[10] MPEG-DASH. Dynamic adaptive streaming over http (dash) – part 6: Dash with server push and websockets. https://www.iso.org/standard/71072.html.

[11] F. Qian et al. Optimizing 360 video delivery over cellular networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, pages 1–6, 2016.

[12] S. Wei et al. Low latency live video streaming over http 2.0. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, pages 37:37–37:42, 2014.

[13] M. Xiao et al. Dash2m: Exploring http/2 for internet streaming to mobile devices. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 22–31, 2016.